REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
10-MAR-2003	Conference Proceedings, (refereed)	

4. TITLE AND SUBTITLE
The Object Event Calculus And Temporal GIS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
0602435N

6. AUTHOR(S)
T. Schmidt FREDERICK E PETRY ROY VICTOR LADNER

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
74-6731-02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Research Laboratory
Marine Geoscience Division
Stennis Space Center, MS 39529-5004

8. REPORTING ORGANIZATION REPORT NUMBER
NRL/PP/7440--03-1011

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217

10. SPONSOR/MONITOR'S ACRONYM(S)
ONR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release,distribution is unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Extension of the temporal database model into the object-oriented database paradigm requires re-assessment of the temporal issues familiar in the RDBMS world. For example, the debate on timestamping relations or individual tuples is mirrored in the question of at what level to apply a timestamp to a temporal object. A brief review of temporal issues as applied to object-oriented databases is supplied.

In our approach, the Event Calculus is used to provide a formalism that avoids the question of object timestamping by not applying time to objects. Rather, temporal behavior is reflected in events, which bring about changes in objects. Previous applications of the Event Calculus in databases are considered. An extension of the formalism to a fully bitemporal model is demonstrated. These extensions and the Object Event Calculus (OEC) form a framework for approaching temporal issues in object-oriented systems. Practical application issues, as well as formal theory are described; an implementation in Java code for the Event Calculus is discussed.

Current GISes will support areal calculations on geographic objects, and can also describe topological relations between them. However, they lack the ability to extrapolate from historical data. The sufficiency of the temporal GIS model to support inventory, updates, quality control and display is demonstrated. Follow-up and further extensions and areas of exploration are presented at the conclusion.

15. SUBJECT TERMS
Event Calculus, Temporal GIS, spatiotemporal data, temporal object model

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Fred Petry
unclassified	unclassified	unclassified	Unlimited	10	19b. TELEPHONE NUMBER *(Include area code)* 228-688-4948

Standard Form 298 (Rev. 8/98)

THE OBJECT EVENT CALCULUS AND TEMPORAL GEOGRAPHIC INFORMATION SYSTEMS

THOMAS M. SCHMIDT
email: dr_thomas_m_schmidt@yahoo.com

Frederick E. Petry and Roy Ladner
Naval Research Laboratory
Mapping, Charting and Geodesy
Stennis Space Center, MS 39529 USA
(rladner,fpetry)@nrlssc.navy.mil

ABSTRACT:

Extension of the temporal database model into the object-oriented database paradigm requires re-assessment of the temporal issues familiar in the RDBMS world. For example, the debate on timestamping relations or individual tuples is mirrored in the question of at what level to apply a timestamp to a temporal object. A brief review of temporal issues as applied to object-oriented databases is supplied.

In our approach, the Event Calculus is used to provide a formalism that avoids the question of object timestamping by not applying time to objects. Rather, temporal behavior is reflected in events, which bring about changes in objects. Previous applications of the Event Calculus in databases are considered. An extension of the formalism to a fully bitemporal model is demonstrated. These extensions and the Object Event Calculus (OEC) form a framework for approaching temporal issues in object-oriented systems. Practical application issues, as well as formal theory are described; an implementation in Java code for the Event Calculus is discussed.

Current GISes will support areal calculations on geographic objects, and can also describe topological relations between them. However, they lack the ability to extrapolate from historical data. The sufficiency of the temporal GIS model to support inventory, updates, quality control and display is demonstrated. Follow-up and further extensions and areas of exploration are presented at the conclusion.

Keywords: Event Calculus, Temporal GIS, spatiotemporal data, temporal object model

INTRODUCTION

Geographical Information Systems (GIS) are repositories of data that relate directly to the physical world. Like conventional databases, they are responsible for storing thematic data about objects in the world. In addition to this role, GISes store spatial data and must implement functions to display the features stored within them. GISes also manage topological relations.

However, as models of the world, GISes have suffered from the lack of a temporal component. Gail Langran's 1989 thesis and subsequent book, *Time in Geographical Information Systems*, [1] offer the first comprehensive survey of the field that has subsequently become known as the study of spatiotemporal data. Previous work included that on cadastral databases that need to maintain an inventory of every owner of a particular piece of property going back to the establishment of the property recording system [2].

There are, however, a number of issues to address before any Temporal GIS (TGIS) can be put into production. The proper modeling of time and capture of the information necessary to reflect accurately what the state of the world and the state of the database is is not a simple problem. There are numerous temporal models, with most referencing or basing themselves on Allen's notion of temporal intervals. branching from the common starting point of Allen's seminal papers in the field of temporal intervals [3,4]. Beyond the question of representing temporality, we also need to consider what underlying database engines can best support GIS queries.

The field of temporal database research has been the focus of much attention in the database community. Introduction to basic concepts is provided in [5,6]. The first comprehensive survey of the field in book format was Tansel's[7]. Advances since 1993 are explicated in [8],

which shows the growing interest in non-relational temporal modeling.

Some basic temporal concepts necessary to an understanding of this enterprise (see [5,6]) include the description of time in a database. Databases may be atemporal, that is, lacking explicit systematic support for temporality, as the original relational model was. They may track the change of the real world; this model of time is called valid time (or world time) and a database that supports it is an historical database. Databases may track transaction time (or system time) and support histories of objects as changed in the database. Such databases are rollback databases. Finally, databases that combine support for transaction and valid time are termed bitemporal. Current usage in temporal database research substitutes temporal where bitemporal is meant.

Early attempts to apply temporal concepts in an object-oriented database model are discussed in [7]. Käfer [9] proposes a temporal extension to an object model. Other noteworthy temporal object models are found in [10,11,12]. Worboys [12] is different in that his object model is event based, and he has applied it in the spatiotemporal arena [13,14].

The concepts discussed in [5,6] demonstrate the early focus in temporal database research on timestamping data. Other researchers have taken a different approach, focusing on time not as an attribute of data but of events that change data. Kowalski and Sergot [17] first proposed the Event Calculus as a way of talking about time in logic programming. Kowalski later demonstrated the use of the Event Calculus in database updates [18], a topic well covered for deductive databases by Sripada [19]. Worboys [13,14,15,20] and Peuquet [21] take an event-based approach to spatiotemporal data without using the formalism of the Event Calculus.

Kesim and Sergot [22] combine the Event Calculus with an object-oriented approach. Their Object-based Event Calculus (OEC) extends the Event Calculus with the ability to track objects with changes over time. However, their proposal offers only an historical approach.

The purpose of this paper is to propose extending the OEC to a bitemporal construction. This will allow an object-oriented temporal database to support queries both about the history of the modeled world and what was believed about the world at different points in time. In the first section, we discuss temporal logic and temporal databases in greater depth. We next describe the Event Calculus, and its extension to the Object Event Calculus. We propose our extension to the OEC for bitemporality, and then describe the uses of temporal data in support of GIS applications. Finally, we provide some advantages and limitations of the approach, and directions for further research.

TEMPORAL LOGIC AND DATABASES

Logic has long formed a framework for understanding databases. If one considers the database as a series of assertions about the world at a series of times, then each state of the database directly correlates to a fact in the predicate logic. The sequence of changes (transactions) in the database likewise correspond to the insertions and deletions of facts in a predicate logic system.

Temporal modal logic represents time by means of modal operators (e.g., Past(Owns(Fred, Plot1)), Future(ISBlue(Sky)). It is an extension to classical logic that provides the logic of possibilities. It provides the ability to completely model the state of the world, regardless of event occurrence (e.g. if we discover that Fran also owned Plot1, we can add fact Past(Owns(Fran, Plot1))). (see [23] for a discussion of temporal logic representations)

Temporal modal logic does, however, possess a number of disadvantages. For example, because states are context sensitive, a context change can require a complex revision to the entire database (as noted particularly in [Srip91] and [KoSe86], this is the frame problem). In addition, explicit references to time are difficult to implement. Proof procedures are less efficient than proof procedures for classical logic.

Allen's Temporal Logic [3] was developed as a method to better implement time in logic. Allen defines the following basic terminology:

Facts: truths that represent states of a process, collected on a discrete or continuous basis;
Events: "happenings" that modify the state of the world; considered instantaneous;
Processes: "groupings" of events that modify the state of the world;
Transitions/Mutations: characterize changes in objects caused by events or processes;
Causation: the coupling of facts, events and processes.

Hajnicz [23] has expanded this list to include:

Actions: Events put in motion by a mover;
Strategies: planning of future actions.

Temporal intervals are at the core of the system. Intervals are fundamental and at base not further subdividable. Events occur in one temporal interval. Allen focuses acutely on the relationships between intervals, and defines seven basic relationships that can hold between intervals. (The diagram is taken from [3])

Relation	Symbol	Symbol for	Pictorial

		Inverse Relation	Example
X before Y	<	>	XXX YYY
X equal Y	=	=	XXX YYY
X meets Y	m	mi	XXXYYY
X overlaps Y	o	oi	XXX YYY
X during Y	d	di	XXX YYYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYY

Numerous scholars have proposed extensions to Allen's temporal logic. Some examples include Fuzzy Temporal Logic (really before, just after) for use in flexible querying [24], and Cobb's use of the temporal relationships in the spatial domain to help define topological relationships among spatial features in a GIS. [25]

The Situational calculus (see, e.g. [26]) was developed to expand upon the ideas first raised in the temporal modal logic. Here, global states are explicit parameters of time-varying relationships. Events are state transitions, with the predicate Holds associating relationship and state (e.g. Holds(Possess(Bob, Book1), S0)). Only one event of a given type may occur in a given state.

The combination of an event type and a state constitutes an event token. Preconditions of events can be expressed as integrity constraints on the states. Because states are named by means of previous states, updates using Happens must be in order (e.g. Happens(give(Bob Book1 John) S0), Happens(give(John Book1 Mary) S1)).

The situational calculus has the advantage that implicit relationships are automatically updated. In addition, updates have semantic structure; they relate successive states. Initiation and termination of relationships is accomplished through event descriptions, not through explicit entry of facts into the fact base. The situational calculus shows a number of disadvantages as well. For instance, because of the use of global states, events need to be totally ordered; an event out of order will not be able to refer to the proper state to update it. Because of this ordering, it is not possible to assimilate new information about the past or make a correction to a previous event.

The implementation of time in relational databases is an ongoing process, from pioneering work developed from the logic arena to the recent TSQL2 and SQL3 standards. Understanding of time in databases requires a grasp of a number of concepts that have been generally accepted (see especially the consensus glossary in [5,6]).

When we talk about time in databases, we mean a system that can be linear, branching (linear in the past with multiple paths in the future) or cyclic. Time can be continuous, analogous to the real numbers, dense (analogous to, e.g. the rational numbers) or discrete. Time can be aggregated into temporal sets, intervals and periods, which are sets of intervals. Finally, time may be absolute (July 1, 1997) or relative (two weeks from now).

The building blocks of any time system are the elements that make up the time line. All systems present some image of a time line, which may or may not branch or have cyclic loops. On these time lines, we define an **Instant** as a time point on an underlying time axis. Likewise, a **Time Interval (TI)** is the time between two instants. We now come to understand the most important temporal object, according to Snodgrass, the **Chronon**. This is a non-decomposable TI of some fixed minimal duration. It can be multi-dimensional; for example, it can be the minimal period of time in both the historic and transactional sense. Any activity in a temporal database is understood to take place during the duration of at least one chronon.

As an example of multi-dimensional intervals, consider the two following definitions. A **Spatiotemporal Interval** is a region in n-space where one axis is spatial and all others are temporal. A **Spatiotemporal Element** is a finite union of these. Similar to this but lacking an explicit spatial component is the **Bitemporal Interval**, a region in two-space of valid time and transaction time (defined below).

Two more basic concepts of the temporal relational model can cause problems for those using other approaches. Most significantly, the **Event** is an instantaneous fact, i.e. something occurring at an instant. Finally, we can understand a **Temporal Element** as a finite union of n-dimensional TI's.

Thus, when we speak of **Valid Time**, (also known as historical time) we are describing when a fact becomes true in the modeled reality, while **Transaction Time** keeps track of when a fact is recorded in a database. Databases that support both time types are known as **Bitemporal** databases. A database can implement time using **User-defined Time**, which is an uninterpreted attribute domain of date and time, e.g. "birth day". Nevertheless, to implement temporality in a database, users must use some **Temporal Data Type** (a time representation specifically designed to meet the needs of users, with the same query-level support as DATE) with which they may **Timestamp** some object, (e.g. attribute

value or tuple), and thus associate a time value associated with it.

We are now ready to approach the concepts in a **Temporal Database**. Using these definitions, the following four database models are commonly described. **Snapshot databases** are the conventional relational database, with flat table structures. **Historical databases** are databases whose underlying relation tracks valid-time; they are meant to track changes to object in the world. Rollback databases are databases whose underlying relation tracks transaction-time, and store every transaction in the database system. Bitemporal databases are databases whose underlying model is bitemporal, and thus a composite of transaction and valid time.

Models for integration of temporal data into the object-oriented arena have received recent attention (see [10] for a more complete recent survey). OODAPLEX, an object-oriented extension to the DAPLEX model, has been extended to support temporal data [7]. Story and Worboys have attempted to detail a class structure for supporting time on a system-wide basis in an object-oriented system. [12]

In contrast to the timestamping approach, several researchers have pursued an event-based approach to time in databases. Peuquet [21] builds a raster-based GIS storing events as changes to a base state. Claramunt [27] applies events in a vector-based GIS, while Worboys [12] adopts the view of change to databases occurring entirely through events without other update mechanisms. All event-based approaches build in some way on the Event Calculus, developed from the Situational Calculus by Kowalski and Sergot[17]

THE EVENT CALCULUS

The situational calculus was designed as a logical programming construct to allow for hypothetical planning based on logic programs augmented with a temporal logic. Situational calculus as a formalism, however, did not allow narrative approaches to time, e.g. what happened, when and what caused certain events to occur.

Kowalski's Event Calculus [17] is a formalism that extends logical reasoning about time to allow for narrative construction. Facts do not become evident in a database or logical system until engendered by events. Events, at least in original formalization, are fundamental and unchangeable; once an event is entered into a logical construct, it can have its resulting conclusions overridden by subsequent events, but it remains as part of the reasoning environment.

Event descriptions are used to describe the existence as well as the beginning and end of time periods. Time periods are determined by relationships which hold during time periods, and the events which can initiate and/or terminate these periods. One event can determine numerous time periods; for example, the event "Alice sold the farm to Bob" (event E1) necessarily defines two time periods: before(E1,has(Alice, Farm)), with has(Alice, Farm) the relationship, defines the time period terminated by event E1; after(E1, has(Bob, Farm)) defines the time period initiated by event E1.

The atom Holds(R1,TP1) means that the relationship R1 holds in the time period TP1. The atom Holds(R1, before(E1,R1)) means that the relationship R1 holds in the period before event E1. Were the term after(E1,R1), then we would know that relationship R1 holds after event E1.

Let us now review the formal axioms for the event calculus, as defined in [17] and refined in [19]. For discussion purposes, we will assume that events whose index is lower occur before events whose index is higher; thus, event E35 occurs before event E76.

Let E1 be an event where Alice sells a farm to Bob. Let event E3 be an event where Bob sells the farm to Charlie. In order for Alice to sell a farm to Bob, she must first possess it; for Bob to sell it to Charlie at event E3 he must possess it for some time before event E3. Event E1 terminates the time period in which Alice owns the farm, and initiates the time period in which Bob owns it. Event E3 terminates the time period of Bob's possession, and initiates the time period of Charlie's possession. Stated formally:

Terminates(e,r) →Holds(r, before(e,r))
Initiates(e,r) → Holds(r, after(e,r))

We now know four distinct time periods, defined by the two events we know about: before(E1, has(Alice, Farm)); after(E1, has(Bob, Farm)); before(E3, has(Bob, Farm)); after(E3, has(Charlie, Farm)). We do not know if after(E1, has(Bob, Farm)) is terminated by E3, we only know that it cannot be in force after E3.

We now wish to discuss a method of describing the start and end of time periods. When we use the atom End(p, e) we mean that time period p has its end with event e. The atom Start(p, e) denotes that the time period p has its start with event e. From this we describe two axioms:

Start(after(e,r), e)
End(before(e,r), e)

Using Logically reasoning from these axioms, it is possible to conclude that the period after(E1,has(Bob, Farm)) is started by event E1. The period before(E1, has(Alice, Farm)) is ended by the event E1.

We now wish to try to specify when a time period is the same. We do not know if the time period after(E1, has(Bob, Farm)) is the same as the time period before(E3, has(Bob, Farm)). We can use the atom Same(t1, t2) to mean that time periods t1 and t2 are identical.

Same(after(e^i, r), before(e^n, r)) → Start(before(e^n, r), e^i)
Same(after(e^i, r), before(e^n, r)) → End(after(e^i, r), e^n)

Using these two axioms, we can state that the period after(E1, has(Bob, Farm)) is terminated by event E3, provided that after(E1, has(Bob, Farm)) and before(E3, has(Bob, Farm)) are the same.

Negation by failure means that the failure to demonstrate a fact from the predicates present in the system means that the fact is proved false; this is an example of the closed-world assumption. Here, because we cannot prove from the predicates that Bob did not possess the farm, we can state that he did. We can state this formally as:

Same(after(e^i, r), before(e^n, r)) ←
Holds(r, after(e^i, r)),
Holds(r, before(e^n, r)),
i < n,
NOT Clipped(e^i, r, e^n)

The predicate Clipped(e^i, r, e^n) means that relationship r, which is initiated by e^i and terminated by e^n, is not a continuous relationship between e^i and e^n. When an event clips a relationship, it means that it instantiates a relationship that is exclusive with a relationship that was known to exist before (or after, if the event terminates a relationship) the event. For instance, if Bob received the farm at event E1, and Bob sold the farm to Charlie at event E5, and event E3 initiates the time period before(E3, has(Dave, Farm)), we know that the relationship in event E3 implies that some event before E3 (presumably E2) terminated the relationship has(Bob, Farm) and some event after E2 and before E5 initiated the relationship has(Bob, Farm); relationship has(Bob, Farm) is clipped between E1 and E5 (in our example, again, presumably by E2). A relationship is always exclusive with itself, and a relationship r1 is exclusive with r2 if and only if it is incompatible with r2; e.g. has(Bob, Farm) is incompatible with has(Alice, Farm). Clipped in axiomatic form is:

Clipped(e^i, r^i, e^n) ← Holds(r^j, after(e^j, r^j)),
Exclusive(r^i, r^j),
e^i < e^j
e^j < e^n
Clipped(e^i, r^i, e^n) ←Holds(r^j, before(e^j, r^j)),
Exclusive(r^i, r^j),
e^i < e^j,
e^j < e^n

From our initial example, we have the time period before(E1, has(Alice, Farm)) and before(E3, has(Bob, Farm)). There is an unknown event that brings about two mutually exclusive relationships. We can try to reason about this incomplete information. Use the function startpoint to return the event that generates the start point of a time period, and the function endpoint to return the event that generates the end point of a time period. Then:

[Start(before(e^j, r^j), startpoint(before(e^j, r^j)))
and
e^i <= startpoint(before(e^j,r^j))] ←
Holds(r^i, before(e^i, r^i)),
Holds(r^j, before(e^j, r^j)),
Exclusive(r^i, r^j),
e^i < e^j,
NOT Clipped(e^i, r^j, e^j)

Now consider the time periods after(E1, has(Alice, Farm)) and after(E5, has(Alice, Farm)). In the first case, the terminating event that caused Alice to lose the farm and regain it at event E5 is unknown; in the second case, the end of the period after(E1, has(Alice, Farm)) is unknown. The following axiom formalizes this notion:

[End(after(e^i, r^i), endpoint(after(e^i, r^i)))
and
endpoint(after(e^i, r^i))] =< e^j ←
Holds(r^i, before(e^i, r^i)),
Holds(r^j, before(e^j, r^j)),
Exclusive(r^i, r^j),
e^i < e^j,
NOT Clipped(e^i, r^j, e^j)

The final uncertainty case we examine is the situation where the end of one time period and the beginning of another time period are unknown. We can infer events that cause the situation to change. For example, after(E1, has(Alice, Farm)) and before(E3, has(Bob, Farm)) are incompatible. We can state this more formally and generally as:

[endpoint(after(e^i, r^i)) <= startpoint(before(e^j, r^j))
and Start(before(e^j, r^j), startpoint(before(e^j, r^j)))
and End(after(e^i, r^i), endpoint(after(e^i, r^i)))]
←
Holds(r^i, before(e^i, r^i)),
Holds(r^j, before(e^j, r^j)),
Incompatible(r^i, r^j),
e^i < e^j,
NOT Clipped(e^i, r^j, e^j)

Now that we have formal axioms for determining if a relationship holds for a certain time period, we wish to describe if a relationship holds at any given time point. The following four axioms allow us to do so.

HoldsAt(r,t) ← Holds(r,p), In(t,p)

In other words, a relationship holds at a time t is it holds for a period p, and the time t is in period p. We define in as follows:

$In(t^j,p)$ ← $Start(p, e^i)$, $End(p,e^k)$, $Time(e^i, t^i)$, $Time(e^k, t^k)$, $t^i < t^j$, $t^j < t^k$

$In(t^j,p)$ ← $Start(p, e^i)$, $Time(e^i, t^i)$, $t^i < t^j$, NOT EXIST e^k $End(p, e^k)$

$In(t^j,p)$ ← $End(p, e^k)$, $Time(e^k, t^k)$, $t^j < t^k$, NOT EXIST e^i $Start(p, e^i)$

These are the 16 basic axioms of the Event Calculus as initially described by Kowalski and Sergot. As Sripada notes "the Event Calculus formalizes a treatment of valid time in historical databases." [19] Provision is not made in the original Event Calculus for revisions to events, and so it cannot accommodate a bitemporal database.

THE OBJECT EVENT CALCULUS

Kesim and Sergot [22] have described a partial extension of the Event Calculus to the object-oriented domain. The model is incomplete in that it does not treat time symmetrically into past and future, and also fails to properly account for bitemporality. The OEC is designed by [KeSe] as an historical database.

As in the Event Calculus, an event initiates a period of time during which some property is true about an object, be it an attribute of the object or simply object existence. The predicate initiates is defined as **initiates(Event, Object, Attribute, Value)**. For example, the act of selling a farm to Bob from Alice initiates a period in which Bob owns the farm: initiates(buys(bob, farm), bob, ownsfarm, true). Its counterpart terminates has the same variable list: **terminates(Event, Object, Attribute, Value)**. In addition to the creation of facts in the database via initiates, OEC provides a method for keeping a journal of events: **happens(Event, Ts)**. Using happens and initiates, it is possible to determine the value of an object at a specific point in time with the holds_at predicate.

holds_at(Obj, Attr, Val, T) ←
happens(Ev, Ts), Ts <= T,
initiates(Ev, Obj, Attr, Val),
not broken(Obj, Attr, Val, Ts, T)

broken(Obj, Attr, Val, Ts, T) ←
happens(Ev*, T*),
Ts < T* <= T,
terminates(Ev*, Obj, Attr, Val)

Thus the query holds_at(bob, ownsfarm, Val, Jan 30 1999 11:00:00) returns the value true for the variable in the query. Holds_for calculates time periods in like fashion. It is defined as:

holds_for(Obj, Attr, Val, Ts-Te) ←
happens(Ev, Ts),
initiates(Ev, Obj, Attr, Val),
terminated(Obj, Attr, Val, Ts, Te)

terminated(Obj, Attr, Val, Ts, Te) ←
happens(Ev, Te), Ts < Te,
terminates(Ev, Obj, Attr, Val),
not broken(Obj, Attr, Val, Ts, Te)

To allow for states to persist when there is no terminating event, the holds_for predicate also can be defined with **not terminated_later**(Obj, Attr, Val, Ts) replacing the terminated clause. Terminated_later is defined as:

terminated_later(Obj, Attr, Val, Ts) ←
happens(Ev, Te), Ts < Te,
terminates(Ev, Obj, Attr, Val)

OEC creates objects by assigning them to a chosen class and then specifying that object's initial state. To determine if an object exists, the instance_of class membership describes a time-varying relationship: if the object is an instance_of a class at a particular time, then the object exists. To specify which events assign objects to new classes, Kesim and Sergot define a new predicate, **assigns.** This is defined as **assigns(Event, Object, Class)** and has a counterpart for object destruction. This is **destroys(Event, Object)** and it is used to delete objects from the system.

Note that the Event Calculus' need to talk about time in relation to events, and events creating the timeline, has been superceded by the Timestamping of events via the happens() predicate. We can now refer to time independently of events in relation to the ordinal years. Thus, while we preserve the ability to refer to time periods in terms of before and after, we can more generally refer to the timeline established by the granularity of time with which we have chosen to stamp events.

EXTENDING THE OEC FOR BITEMPORALITY

Because events in the OEC must be stored in the order that they occur in the real world, the database modeled, while tracking historical time, actually follows the implementation of a transaction-time temporal model. Recall that a transaction-time model cannot store events (transactions) in any order other than the system timestamp order; this is the order in which the database receives the transactions and any other is inconceivable. Indeed, as Kesim and Sergot [19] note, "if valid times and transaction times are distinguished but are exactly correlated, then [their model] can be seen as a 'degenerate bitemporal' database."

Kesim and Sergot implicitly implement a distinction between transaction-time and valid-time events. They note that they "have introduced two separate sets of predicates, one for dealing with change in internal state of objects and one for creation/deletion of objects." They do this because they "want to emphasize the conceptual difference between changes in an object's state ... and changes to class membership." What they have really done is delineate the need to define predicates for maintenance of real-world objects (the changes in objects' state) and for maintenance of database objects (object creation/destruction, event revision).

Kesim and Sergot do face the problem of correct description of the world. What if an event that alters or creates an object has been recorded incorrectly? The database will always reflect the most recent understanding of the event. This is done by storing events in a "journal"; because events are objects in the OEC, this journal must be a database of event objects. Because revisions to knowledge about the real world are permitted, the database of event objects is permitted to be updated to reflect new information. The new event object supercedes the old event object and completely replaces it; neither the event that changed the event object, nor the prior version of the event object is retained.

We do not actually need separate predicates for real-world and database events, however. We can extend the real-world OEC predicates to apply to the "world" of the journal. Thus creation of an event that changes a real world object will result in at least one invocation of the assigns() predicate, to create the event object that changes the real-world object. If the event being added to the journal also creates a real-world object, then the assigns predicate is used to again to create it.

The journal of events described is, again, simply a collection of objects. Changes to these objects, and changes to the database itself, are, as Sripada noted, meta-events [17]. Sripada proposes the revises(E', E) predicate to reflect revisions to events. The revises() predicate, while claimed of use by Sripada in an historical database, in fact implements a type of rollback database. We thus adopt it for use in the OEC, restricting the domain of objects upon which the predicate operates to the database of event objects, the journal.

We need to describe when revision to previous objects occurs. Happens(Ev, Ts) relates an event's occurrence to the timeline we have chosen for our implementation. Likewise, we define **happens_t(Ev, Ts)** as a predicate, to timestamp the occurrence of transaction-time events.

When we examine the question of a revision to an event object, we will again assign an event object with the assigns predicate. Now we will also specify the object revised and the revisions to the object; this requires the use of the revises() predicate we adopted from Sripada. Since the event to be changed is an object, the revision event also initiates and terminates attribute values about certain objects, in this case already extant events.

We now have a construct that allows a fully bitemporal view of the world. Since we timestamp events and meta-events, we can determine along two time axes the state of the world. We can now answer the following types of questions: What was the state of the world on Jan 1 2000? What did we believe was the state of the world on Jan 5 2000? On Jan 5 2000, what did we believe was the state of the world on Jan 1 2000? What changed about the world, and our belief about the world, between Jan 1 and Jan 10 2000?

APPLICABILITY TO TEMPORAL GIS

Langran offers the following six justifications for a temporal GIS:

Inventory: a temporal GIS should be able to store the most complete possible description of the study area;

Analysis: since atemporal databases do not store changes, they provide no facility to examine processes that cause changes to features in the GIS;

Scheduling: a temporal GIS should allow for the user to set intervention points in the system;

Display: a temporal GIS should allow better displays of time-series data like the growth rings of a city like Chicago;

Updates: rather than distribute an entirely new copy of the database, a temporal GIS could send only the updates;

Quality Control: a temporal GIS could store the lineage of each item in it, allowing for better management of accuracy in data.

In addition to these six proposed by Langran, a temporal GIS should assist with at least two other issues:

Conflation: the process of combining two maps from different sources;

Uncertainty: it is not simple to represent uncertain and incomplete information correctly in current GISes.

Without the ability to answer questions in all eight of the areas outlined above, a GIS cannot be considered complete. The construction of a proper temporal GIS (TGIS) will require that functionality in the TGIS support each of these eight requirements. In supporting these requirements, the TGIS will need to offer a proper representation for time in the GIS. Likewise, the TGIS should support the thematic and spatial data in a storage structure that appropriately matches the needs of both of these types of data, just like any non-temporal GIS.

The OEC approach allows the underlying database to support each of Langran's requirements. Recording and storing all changes to objects allows for thorough inventory control. Distributing only the event objects that change the database will reduce the volume of data transmitted in, say, a CORBA over IIOP environment. Finally, as discussed in reference to the Event Calculus, we have a mechanism for reasoning about uncertain information.

CONCLUSION

The bitemporal OEC displays the advantages of all event-based temporal models: chiefly, time is not an inherent attribute of objects, but an attribute of events that occur at specific points or over specific periods. As an extension to the OEC, it allows for full modeling of temporal information about objects in the real world and the database. Because temporal support maintains a more complete history of the world, information that was previously destroyed on updates is preserved. This is advantageous in a wide range of applications, with particular focus on GISes.

The chief limitations of temporal approaches are in the volume of data needed to be stored, and the overhead in data retrieval. Several articles in [8] discuss optimization strategies for temporal queries. Revisions in a logic-based approach like the Event Calculus must necessarily require regeneration of all subsequent conclusions. This overhead has led most researchers to limit the ability to revise existing events. Query optimization techniques for deductive databases, like those presented by Nussbaum in [28] can allow the addition of event revisions.

Further work in the bitemporal OEC is suggested herein. We are pursuing an implementation in Java with persistent storage supplied by Objectstore. We intend to use this model to support a Temporal GIS.

Other applications include systems that need to combine reasoning abilities with persistent data storage. Production and rule-based systems can benefit from an event-oriented approach, as they will be able to determine not only courses of action, but the reasons why previous courses of action have been undertaken.

Finally, as a competing model to the timestamping paradigm for temporal databases, the OEC can supply the underlying engine where proposals call for a temporal database.

ACKNOWLEDGEMENTS

We would like to thank the Naval Research Laboratory's Base Program, Program Element No. 0602435N for sponsoring this research.

REFERENCES

[1] Langran, Gail. *Time in Geographic Information Systems,* Bristol, PA: Taylor and Francis, 1992.

[2] Basoglu, U. "The Efficient Hierarchical Data Structure for the US Historical Boundary File," *Harvard Papers on Geographical Information Systems,* 4: 1978.

[3] Allen, J. "Maintaining Knowledge about Temporal Intervals," *CACM* November 1983: 832.

[4] Allen, J., "Towards a general theory of action and time," *Artificial Intelligence* V.23 #2, pp. 123-54

[5] Jensen, C., Clifford, J., Elmasri, R., Gadia, S., et al., "A consensus glossary of temporal database concepts," *SIGMOD Record* V. 23, #1

[6] Jensen, C.S.; et al. "The consensus glossary of temporal database concepts-February 1998 version," in [8].

[7] Tansel, Abdullah Uz. *Temporal Databases: Theory, Design and Implementation.* New York: Benjamin/Cummings, 1993.

[8] Etzion, O.; Jajodia, S.; Sripada, S. *Temporal Databases: Research and Practice.* Berlin: Springer-Verlag; 1998.

[9] Käfer, W., Ritter, N., Schöning, H., "Support for Temporal Data by Complex Objects", *Proceedings of the 16th VLDB Conference*, Brisbane, Australia. New York: Springer-Verlag, 1991.

[10] Bertino, E., Ferrari, E. and Guerrini, G. "A Formal Temporal Object-Oriented Data Model," *International Conference on Extending Database Technology,* Avignon, 1996.

[11] Su SYW; Hyun SJ; Chen HHM. "Temporal Association Algebra - A Mathematical Foundation for Processing Object-Oriented Temporal Databases," *IEEE Transactions on Knowledge and Data Engineering* 1998, Vol 10, Iss 3.

[12] Story, P., and Worboys, M. "An object-oriented model of time," *Technical Report TR95-03.* Keele, Staffordshire, UK: Department of Computer Science, Keele University.

[13] Worboys, M. "A Unified Model for Spatial and Temporal Information," *The Computer Journal,* 37: 1, 1994.

[14] Worboys, M. "Object-Oriented Approaches to Geo-Referenced Information," *International Journal of Geographical Information Systems,* 8: 4, 1994.

[15] Worboys, M. "A Generic Model for Spatio-Bitemporal Geographic Information," in [16], pp 25-39.

[16] Egenhofer, M. and Golledge, R. *Spatial and Temporal Reasoning in Geographic Information Systems.* New York: Oxford University Press; 1998.

[17] Kowalski, R., and Sergot, M. "A Logic-Based Calculus of Events," *New Generation Computing,* 4: 1, 1986.

[18] Kowalski, R. "Database Updates in the Event Calculus," *Journal of Logic Programming,* 12, 121-146, 1992.

[19] Sripada, S. *Temporal Reasoning in Deductive Databases.* Doctoral Dissertation, University of London, 1991.

[20] Worboys, M., Hearnshaw, H., and Maguire, D., "Object-Oriented Data Modelling for Spatial Databases", *International Journal of Geographical Information Systems*, Vol. 4, No. 4, 1994.

[21] Peuquet, D. "An Event-Based Spatiotemporal Model for Temporal Analysis of Geographical Data," *International Journal of Geographical Information Systems,* 9: 1, 1995.

[22] Kesim, F., and Sergot, M. "A Logic Programming Framework for Modelling Temporal Objects," *IEEE Transactions on Knowledge and Data Engineering*, 8:5 1996, 724-741.

[23] Hajnicz, Elzbieta. *Time Structures: Formal Description and Algorithmic Representation.* New York: Springer, 1996.

[24] Bose, P, Connan, F, and Rocacher, D "Flexible Querying and Temporal Databases", 1997

[25] Cobb, M., Ph.D. thesis, Tulane University, 1995.

[26] Kowalski, R., and Sadri, F. "The Situational Calculus and Event Calculus Compared," *Proceedings of the International Symposium on Logic Programming*, pp. 539-553, 1994.

[27] Claramunt, C., and Theriault, M. "Managing Time in GIS: An Event-Oriented Approach," *Recent Advances in Temporal Databases,* Clifford, 1995.

[28] Nussbaum, M. Building a Deductive Database. Norwood, NJ: Ablex Publishing Corporation; 1992.